

High Speed Energy Efficient Non-Strobed Sensing for Nano-scale SRAMs

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Master of Science (Electrical and Computer Engineering)

by

Sudhanshu Khanna

August 2011

APPROVAL SHEET

The thesis is submitted in partial fulfillment of the

requirements for the degree of

Master of Science (Electrical and Computer Engineering)

Sudhanshu Khanna

This thesis has been read and approved by the examining Committee:

Benton H. Calhoun (Thesis advisor)

Mircea R. Stan (Chair)

John Lach

Accepted for the School of Engineering and Applied Science:

Kathryn C. Thornton (Dean, School of Engineering and Applied Science)

August 2011

Abstract

Static Random Access Memory (SRAM) is the most common form of storage on modern day SoCs. It brings together advantages of high speed, low power, low area, and compatibility with conventional logic CMOS process technologies. SRAM structures often occupy 50-75% of the area in today's SoCs, and are often on the timing critical path. Thus, any reductions in area, power, or delay improve chip level metrics dramatically. For the past two decades, technology scaling has helped reduce area, improve performance and lower power. However, leakage and local parameter variation are the two big issues we face in the deep submicron regime. As a result of leakage and local variation, technology scaling no longer results in an automatic decrease in power and performance. In particular, SRAM bit-cells which are composed for near-minimum sized transistors are impacted the most by local variation. This thesis presents ideas and implementations to help improve SRAM read access performance and energy per read access in the presence of variations. We present a novel pipelined sensing scheme using non-strobed sense amplifiers. The scheme helps in achieving higher SRAM read throughput while also reducing energy per read operation. We show why the use of non-strobed sense amplifiers is more beneficial than conventional strobed sense amplifiers while pipelining an SRAM. Simulation results in a commercial 45nm technology node are presented.

Conventional SRAM timing is based on worst case design. Systems are designed with enough timing margin such that the worst case bit-cell can be correctly read. In the face of growing variation, this paradigm results in excessive pessimism. We present a new sensing scheme for asynchronous SRAMs which help move away from the worst case paradigm, and has a read access delay that depends on the bit-cell being read. To make this adaptive scheme

possible, we propose a novel differential non-strobed sense amplifier that helps improve average read access time, reduces the energy per read operation, and is more robust than single ended non-strobed sense amplifiers. The proposed sense amplifier can also be used in synchronous SRAMs. Simulation results in a commercial 45nm technology node are presented.

Acknowledgements

I would like to thank my advisor, Prof. Benton H. Calhoun, for giving me the opportunity to work with him and for his encouragement and support throughout my academic work at University of Virginia. I would also like to thank Prof. John Lach, Prof Mircea Stan, and Prof Ronald Williams for useful discussions without which this thesis would not have been possible. I want to thank my colleagues and collaborators Satyanand Nalam, Anurag Nigam, Joesph Ryan, Randy Mann, Kyle Craig, Yousef Shasksheer, Jiajing Wang, and Saad Arrabi for interesting discussions and a wonderful college experience. I also want to thank my family for their support and belief in me; I could not have done this without you.

Table of Contents

Abstract	3
Acknowledgements	5
Chapter 1: Introduction: SRAM design in the presence of local variations	7
1.1. <i>Motivation</i>	7
1.2. <i>Contributions of this thesis</i>	8
1.3. <i>Outline of this thesis</i>	9
1.4. <i>Terminology</i>	10
Chapter 2: A Fast Low Power Non-Strobed Pipelined Sensing Scheme for Nano-scale SRAMs	12
2.1. <i>Introduction</i>	12
2.2. <i>Pipelining in SRAMs using strobed sensing</i>	16
2.3. <i>Timing in non-strobed sensing & modified timing for pipelining</i>	19
2.4. <i>Non-strobed pipelined sensing scheme: Circuit implementation and Results</i>	26
2.5. <i>Conclusion</i>	32
Chapter 3: A Fast Low Power Non-Strobed Differential Sense Amplifier and Sensing Scheme for Asynchronous SRAMs	33
3.1. <i>Introduction</i>	33
3.2. <i>Timing in strobed and non-strobed sensing schemes</i>	35
3.3. <i>Differential Non-Strobed Sense Amplifier</i>	39
3.4. <i>Asynchronous SRAM with DNSA</i>	43
3.5. <i>Conclusion</i>	47
Chapter 4: Conclusion and Future Work	48
4.1. <i>Summary</i>	48
4.2. <i>Contributions</i>	49
4.3. <i>Future Work</i>	50
Bibliography	52
List of Publications	54

Chapter 1: Introduction: SRAM design in the presence of local variations

SRAMs are a critical component in all modern digital Integrated Circuits (ICs). With CMOS processing technology entering the deep submicron regime, local variation in device parameters has become the biggest roadblock in SRAM design. As a result of growing variation, technology scaling doesn't result in the larger performance gain in SRAM that was earlier easily achieved. This chapter presents the motivation and outline for the thesis, which presents novel methods and circuits to make SRAMs and processors faster and lower in power consumption. It also provides an outline of this thesis. Finally, it presents a glossary of the terminology used in this thesis.

1.1. Motivation

Technology scaling has been fuelling the growth of the semiconductor industry for the last two decades [1]. The cost of an IC is highly dependent on the size of the die. Thus, as technology scales, the cost per IC decreases dramatically. Moreover, along with lower cost, scaling traditionally also lowered power and improved performance. However, with channel lengths moving below the 90nm range, careful control over geometries drawn and impurities added during fabrication is extremely challenging. This has manifested in increase local and global device parameter variations. Transistors with small width and lengths are worst impacted by such variations [2]. SRAM bit-cells occupy 50-75% of most modern day SoCs. Thus making small sized bit-cells is critical in lowering die area and cost. As a result, bit-cells have near minimum sized transistors, and are worst impacted by local variation. This means that a lot of timing margin has to be put in SRAM design to make sure all of the millions of bit-cells on a chip

actually work. Designing fast, low power SRAMs in the presence of variation is the motivation for this thesis. We explore the use of non-strobed sensing, and propose SRAM pipelining in the first part of the thesis. In the second part, we propose a novel differential non-strobed sense amplifier to enable the use of adaptive sensing in asynchronous SRAMs. Adaptive sensing helps move away from the worst case paradigm in SRAM timing. The differential non-strobed sense amplifier can be used in synchronous SRAMs as well.

1.2. Contributions of this thesis

Drawing from our designs and analysis, this thesis makes the following key contributions:

- Shows why the long BL droop development phase limits the benefit of pipelining an SRAM (Sections 2.1 and 2.2).
- Shows for the first time how the unique characteristics of a non-strobed sense amplifier can be used to drastically reduce the BL droop development phase at the cost of higher SA delay. This is nothing but a knob that helps balance stages of a pipeline, thereby maximizing the benefit of pipelining an SRAM (Section 2.3).
- Demonstrates the Non-Strobed Pipelined Sensing (NSPS) scheme. NSPS helps increase performance by 1.6X as compared to a non-pipelined strobed SRAM, while also decreasing energy per read by 28%. As compared to a 2-stage pipelined strobed SRAM, NSPS is 18% faster (Section 2.4).
- Shows that conventional (strobed) sensing can only be designed with a worst case timing scheme and thus cannot be used to design an (asynchronous or synchronous) SRAM whose delay depends on the particular bit-cell being read (Section 3.2).

- Demonstrates a sensing scheme using a novel differential non-strobed sense amplifier to enable local-variation-adaptive SRAM delay. The technique improves asynchronous SRAM average read delay by up to 17%. The differential non-strobed reduces energy per read operation by 8.5% and is more robust than single ended non-strobed sense amplifiers. The proposed sense amplifier can also be used in synchronous SRAMs (Sections 3.3 and 3.4).

1.3. Outline of this thesis

- **Chapter 1 – Introduction: SRAM design in the presence of variation:** This chapter lays out the motivation and contributions of this thesis. It also provides a summary of the terminology used, and provides an outline for this work.
- **Chapter 2 – A Fast Low Power Non-Strobed Pipelined Sensing Scheme for Nano-scale SRAMs:** This chapter proposes the use of pipelining to increase the throughput of instruction caches. We use non-strobed sensing to improve pipelining efficiency, performance and energy per operation.
- **Chapter 3 – A Fast Low Power Non-Strobed Differential Sense Amplifier and Sensing Scheme for Asynchronous SRAMs:** This chapter proposes a novel differential non-strobed sense amplifier that enables the implementation of a adaptive sensing scheme in asynchronous SRAMs. The adaptive scheme helps have an SRAM delay that changes with the bit-cell being read, and is not limited by worst case timing. The differential non-strobed sense amplifier is more robust, has higher performance, and lower energy per operation than single ended non-strobed sense amplifiers. The differential non-strobed sense amplifier can be used in synchronous SRAMs as well.

- **Chapter 4 – Conclusion:** Conclusions, contributions of this work and future work are described in this chapter.
- **Bibliography:** A list of references used in this thesis.

1.4. Terminology

Bitcell (cell): Basic unit of an SRAM that stores one bit of data. Essentially composed of a crosscoupled inverter pair and zero or more access ports or transistors.

BL: Bitline, a wire that connects the bitcell, possibly through an access transistor, to the sense amplifiers and the Bitline drivers which supply the data during a write.

BLB: Bitline-bar, a complementary Bitline, present in a conventional 6T bitcell.

CMOS: Complementary MOS, circuits that contain both NMOS and PMOS devices.

DNSA: Differential non-strobed sense amplifier.

Leakage: In this thesis, refers to sub-threshold leakage, which is the current flowing between the drain and source of a transistor when it is in the off-state, that is the gate of the transistor is below the threshold voltage.

MC Simulation: Monte Carlo simulation, the technique of simulating a circuit over a wide range of randomly chosen values for device parameters [3].

MOS(FET): Metal Oxide Semiconductor Field-Effect Transistor, a transistor that uses a metaloxide as an insulator between a polysilicon gate and a semiconductor. An electric field can be used to create an inversion layer or channel between the source and drain terminals of the transistor.

NMOS: A MOSFET that utilizes an n-type inversion layer for conducting current.

NSR-SA: Non-strobed single ended regenerative sense amplifier

PMOS: A MOSFET that utilizes a p-type inversion layer for conducting current.

Sense Amplifier (SA): An analog circuit that amplifies a differential voltage. It is used to speed up reading by sensing and amplify the differential between BL and BLB. It also helps in avoiding the energy overhead of fully discharging the bitlines which have large capacitances.

SRAM: Static Random Access Memory, which stores data statically using a cross-coupled inverter pair.

Transistor: Refers to a MOSFET in this thesis.

VDD: Reference for the high potential power supply (1.0 V in this thesis).

VSS: Ground, reference for the low potential power supply (0 V).

V_T : Threshold voltage, the voltage at which the channel in a transistor undergoes strong inversion and begins conducting.

WL: Wordline, a wire that controls the gates of the access transistors of a bitcell.

Chapter 2: A Fast Low Power Non-Strobed Pipelined Sensing Scheme for Nano-scale SRAMs

This chapter describes a novel Non-Strobed Pipelined Sensing (NSPS) scheme where the read operation of an SRAM is pipelined into two clock cycles. As a result of efficient pipelining, the SRAM can work at 1.6X faster clock frequency as compared to a strobed non-pipelined SRAM. The key to the pipelining scheme is the ability to reduce the BL droop development time at the cost of higher sense amplifier (SA) delay, which is then pipelined into the next clock cycle. Lowering the WL pulse width also helps lower energy per read operation by 28%. This is the first work that demonstrates how BL droop development time can be traded-off with SA delay, enabling efficient SRAM pipelining with little impact on area. We show the concept, circuit implementation, and simulation results in a commercial 45nm technology node.

2.1. Introduction

On-chip SRAMs are a ubiquitous component of modern digital systems ranging from high end server processors to micro-sensor nodes. SRAMs dominate the silicon area of digital processing systems, often accounting more than 50% of die [4]. Thus lowering the bit-cell footprint can substantially reduce die area and cost. Technology scaling helps by lowering area by half every two years [5]. However, as technology scales, local variations in transistor parameters also increase. The small transistors in SRAM bit-cells are worst affected by this variation [2]. With small bit-cells driving the large bit-line (BL) capacitance, BL droop development is usually the longest phase in an SRAM read operation.

Figure 2.1 shows the block diagram of a strobed SRAM, its critical timing arches and the associated waveforms. A popular voltage latched strobed SA with optimized sizing from [6] is used. The delays of the different blocks are measured at the typical corner, and after running 100K montecarlo sims to account for variations in the bit-cell and sense amplifier (SA). 512-row static CMOS decoder is used. SRAM sub-blocks are synchronized by timing signals that are generated using the system clock and programmable delay elements. A delay T_{DEC} after the rising edge of the clock the correct word-line (WL) is generated. The delay T_{DEC} is set by the row decoder delay. Then the WL driver, launches the WL signal across the bit-cell array. Upon the arrival of the WL pulse, the bit-cell starts developing droop on the BL (in case of read-0). Also, for the duration of the WL pulse, the BLs are connected to the sense amplifiers (SA) by the column multiplexor. A delay T_{BC} after the decoder output (the correct WL), the sense amplifier enable (SAE) pulse arrives. T_{BC} is therefore the sum of WL driver delay and the bit-cell delay. Thus, worst case T_{BC} is set such that the weakest bit-cell has enough time to pull the BL below the SA offset. The WL pulse turns off just after the SAE pulse arrives, thus the WL pulse width is very close to T_{BC} . The SA has a finite resolution time, and this sets the width of the SAE pulse, T_{SA} . Note that BL pre-charge happens in parallel with WL decode, and is usually shorter in duration and is not in the critical path.

Pipelining is a technique that has been commonly used in processor design where logic operations are split into multiple cycles to enable faster clock frequencies. However, the maximum clock frequency of a pipeline is limited by the delay of the longest stage. In this chapter we apply pipelining to an SRAM. By introducing a flip-flop (FF) at the input of the WL driver and using the column mux at the boundary of bit-cell and SA as a dynamic transmission gate latch, it is ideally possible to pipeline an SRAM into 3 stages as shown in Figure 2.2.

Addition of a FF not only adds a delay but also makes the WL driver layout very tricky, since it must be pitch mapped to the small sized bit-cell height. Thus, another option while pipelining an SRAM is a 2-stage pipeline where the decoder and BL droop development are in the first stage and SA is in the second stage. Such a 2-stage pipelined SRAM has the same structure as the non-pipelined SRAM of Figure 2.1 as the column mux is anyways needed in the non-pipelined SRAM. For the reason of higher system complexity of using a 3-stage pipelined memory (as discussed later) and the delay/area overhead due to addition of FFs in each WL driver, we will focus on a 2-stage pipelined SRAM, and only provide estimates for the 3-stage case.

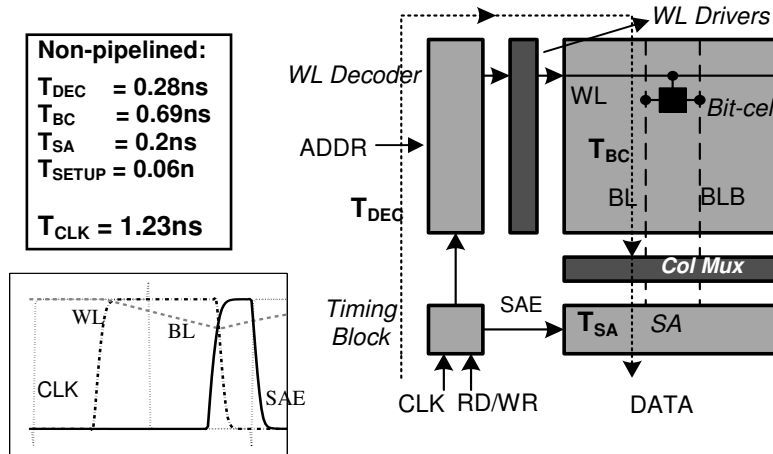


Figure 2.1: Conventional non-pipelined SRAM macro with timing arches (dotted). Timing numbers for typical corner and worst case across 100K MC sims. A 2-stage pipelined SRAM with decoder and BL droop dev in the first stage and SA in the second stage would structurally look the same as a column mux is already present in a non-pipelined SRAM. VDD is 1V.

While it is structurally possible to pipeline an SRAM as described above, pipelining has been used to a much lower extent in SRAM design. A reason for this can be understood by observing the stage delays mentioned in Figure 2.1 and 2.2. Though the SRAM read operation can be split into three/two cycles, BL droop development is a long unbalanced pipeline stage which limits the maximum throughput. This fundamentally lowers efficiency of SRAM pipelining. BL droop

development involves the small signal discharge of the large BL capacitance by the small pass transistor and pull-down transistor of a bit-cell. This is conceptually equivalent to a single gate delay with a large BL capacitance fan-out. This makes BL droop development very difficult to split into two or more steps.

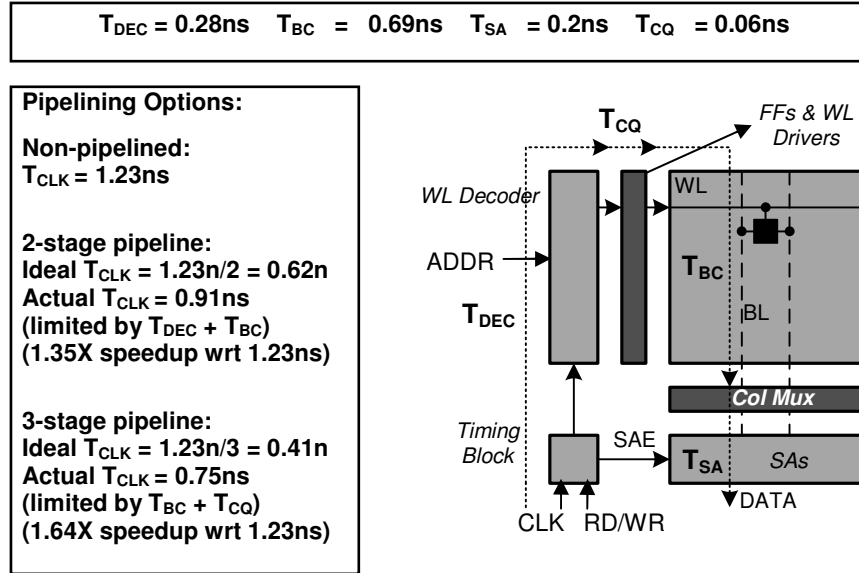


Figure 2.2: 3-stage pipelined SRAM macro with critical timing arches (dotted). Decoder, BL droop dev and SA are separated by the FF and col mux stages. For 2-cycle pipeline timing numbers, decoder and BL droop are combined in the same stage to avoid using flip-flops. 2-stage pipelined SRAM would be structurally same as block diagram in Figure 2.1. Timing numbers are for typical corner and worst case across 100K MC sims. VDD is 1V.

In this work, we use non-strobed sensing that enables a trade-off between the BL droop development time and SA delay. Shortening the long unbalanced BL phase at the cost of SA delay enables us to balance stages of an SRAM and thus efficiently pipeline an SRAM. An SRAM using a strobed SA doesn't have this knob because the strobed SA needs a minimum BL droop dictated by its input referred offset. However, non-strobed sense amplifiers don't have such a strict offset requirement and can trade-off BL droop with SA delay. Hence we call our

technique the Non-Strobed Pipelined Sensing Scheme (NSPS). Drawing from our designs and analysis, this chapter describes the following key contributions:

- Shows why the long BL droop development phase limits the benefit of pipelining an SRAM (Sections 2.1 and 2.2).
- Shows for the first time how the unique characteristics of a non-strobed sense amplifier can be used to drastically reduce the BL droop development phase at the cost of higher SA delay. This is nothing but a knob that helps balance stages of a pipeline, thereby maximizing the benefit of pipelining an SRAM (Section 2.3).
- Demonstrates the Non-Strobed Pipelined Sensing (NSPS) scheme (Section 2.4).

Section 2.2 describes previous work in SRAM pipelining and the system level impact of pipelining. Section 2.3 analyses non-strobed sensing, and its application to NSPS. Section 2.4 describes the Non-Strobed Pipelined Sensing (NSPS) scheme and a 45nm SRAM using NSPS. Section 2.5 concludes this chapter.

2.2. Pipelining in SRAMs using strobed sensing

Looking at the waveforms in Figure 2.1, the read operation of a conventional SRAM has three phases: Decoder delay (clock rising edge to generation of correct WL driver input), sum of WL driver delay and bit-cell delay (WL driver input to SA input), and SA delay. If we pipeline a strobed SRAM of Figure 2.1 into 3 stages as is done in Figure 2.2, with the original clock period at 1.23ns, an ideal pipeline with balanced stages could have reduced the clock period to 0.41ns ($=1.23\text{ns}/3$). However, the long unbalanced BL droop development phase limits the minimum clock period to 0.75ns ($=T_{BC}+T_{CQ}$).

On the other hand, if we were to pipeline the SRAM into two stages (decoder and BL droop in the first stage and SA in the second), the minimum clock period would be 0.97ns, as opposed to ideal value of 0.62ns ($=1.23\text{ns}/2$). Again, this is an inefficient pipeline because of unbalanced stages. Note that we combine decoder and BL droop development into a single stage rather than BL droop development and SA so that we can avoid adding a FF in each WL driver. This reduces the T_{SETUP} and T_{CQ} overheads, but also area overhead in the pitch mapped WL driver. As mentioned before, such a 2-stage SRAM would structurally look exactly like the SRAM in Figure 2.1 as column mux is anyways present in a non-pipelined strobed SRAM. Looking at the 2-stage pipeline timing summary in Figure 2.2, if we could somehow reduce the BL droop development phase (T_{BC}) at the cost of SA delay (T_{SA}), we could balance stages and get closer to the ideal 0.62ns clock period for 2 stages. However, as mentioned in section 2.1, reducing T_{BC} is not possible (without changing the bit-cell or BL length [7], both of which increase SRAM area and power drastically) because a strobed SA needs a minimum BL droop dictated by its input referred offset. Figure 2.2 summarizes the performance of the strobed SRAM in non-pipelined, 2 and 3 stage pipeline versions, including the speedup provided by pipelining.

The work in [7] and [12] presents a pipelined cache architecture based on splitting the SRAM read into three cycles as shown in Figure 2.2. The authors in [7] highlight the long unbalanced BL droop development phase and propose a technique to reduce its duration thereby balancing the three stages. To lower the BL droop development time the authors split the SRAM into smaller banks, thereby reducing BL length (thus capacitance). However, splitting an SRAM into banks results in significant area overhead as the peripheral circuits have to be replicated for each bank. As mentioned before, adding flip-flops at WL driver-bitcell boundaries also adds to the area overhead. In total, the work reports a 32% area overhead which is significant considering

that a large portion a SoC is occupied by SRAMs. In another work [8][3], the read operation of a large multibank 256kB SRAM used in the CELL processor is split into 6 cycles. This work splits decoder of this large SRAM into 3 stages, and the output data forwarding into 2 stages. However, here too the BL droop development remains a single difficult to split stage. On the other hand logic functions like the decoders can ideally be split into arbitrary number of stages. In summary, pipelining is a technique that has not been explored often in the context of SRAM design, with one of the primary reasons being the roadblock of a long unbalanced BL droop development phase. This work shows for the first time how the unique characteristics of a non-strobed sense amplifier can be used to drastically reduce the BL droop development phase (T_{BC}) at the cost of higher SA delay (T_{SA}). This is nothing but a knob that helps balance stages of a pipeline, thereby improving SRAM pipelining efficiency.

The second reason which restricts pipelining in an SRAM is the system level impact. Though pipelining an SRAM results in higher throughput, pipelining has the drawback that the address must be provided to the SRAM by the processor one cycle in advance (for a 2-stage pipeline). This is easily accomplished in the case of instruction memories, where most accesses are sequential in nature. The next address is predictable except in the case when the processor encounters a branch instruction that is taken. In case of a branch instruction that is taken, a processor using a pipelined memory is likely to have one extra stall as compared to the usual number of stalls associated with a taken conditional branch. Thus, unless the speedup due to pipelining is large, the impact of the extra stall restricts the applicability of pipelining an SRAM. The long unbalanced BL droop development phase has till now limited the speedup provided by pipelining a strobed SRAM, but as the following analysis shows, our technique can achieve a much higher speedup that results in an overall win even after accounting for the extra stall.

Across a wide range of benchmarks, the average number of branch instructions is reported to be around 20%, and out of these branches 65% are taken on an average [9]. This means that 13% ($=20*0.65$) of the instructions are branches that are taken. Assuming the worst case scenario that a processor has no stalls for a branch while using a normal instruction cache and one stall for a conditional branch while using the pipelined instruction cache, the CPI while using a pipelined instruction cache would be 1.13 ($=1*0.87+2*0.13$). Figure 2.2 shows that the speedup provided by 2-stage strobed pipelined SRAM is 1.35X. When used in a processor as an instruction cache, this SRAM would have an effective speedup of around 1.2X ($=1.35X/1.13$). We show in the next section our NSPS 2-stage SRAM which achieves a speedup of $\sim 2X$. Following the preceding analysis, if our 2-stage NSPS SRAM is used as an instruction cache, the effective speedup would be 1.7X ($=2X/1.13$).

Complexity in system design is another reason that a 3-stage pipeline is impractical. A 3-stage pipelined SRAM requires that the address be provided to it by the processor 2 cycles in advance. Analysis similar to the 2-stage pipeline results in a CPI of 1.26 for the 3-stage pipelined SRAM. For the reason of higher CPI and area overhead due to addition of flip-flops in each WL driver, we will focus on 2-stage pipelined SRAM, and only provide estimates for the 3-stage case.

2.3. Timing in non-strobed sensing & modified timing for pipelining

As we saw in the last section, the long BL droop development phase is the limiting, difficult to split portion of an SRAM read operation. To balance the SRAM pipeline stages we must try to reduce the duration of this phase (at the cost of other phases), as was done in [7] **Error! Reference source not found.** by splitting the SRAM into smaller banks. However, in [7] this came at a large

area and power penalty. In this section we will describe how non-strobed sensing can help shorten the BL droop development phase at the cost of SA delay, with little impact on area and power. Since BL droop development phase has the same duration as the WL pulse width, we can equivalently say that if the WL pulse width can be made shorter, SRAM pipelining can be made more efficient.

In this section we will first describe non-strobed sensing as proposed in [10] for non-pipelined SRAMs. We will then modify the timing described in [10] by using a truncated WL pulse. A truncated WL pulse means a shorter BL droop development phase, which is what we need to balance SRAM pipeline stages.

2.3.1. Non-Strobed Sensing for Non-pipelined SRAMs

The timing of a non-strobed sensing scheme is significantly different than strobed SRAM timing of Figure 2.2. In this work we use a single ended non-strobed regenerative sense amplifier (NSR-SA) [10], however the concepts described in our work can be applied to other AC-coupled non-strobed sense amplifiers like [11] as well.

The NSR-SA circuit and timing and a non-strobed non-pipelined SRAM column are shown in Figure 2.3. Like the strobed sensing scheme in Figure 2.2, the first phase of the read operation is WL decode (of same duration T_{DEC}). BLs are also pre-charged during WL decode. At the end of the WL decode phase, the WL of the selected row is pulsed high. The NSR-SA has an AC-coupled capacitor (C1) at its input followed by an inverter (M1-M2). This capacitor-inverter pair is then repeated (C2, M3-M4) in a cascade. The important difference from strobed timing is that during WL decode, the switches S_{AZ} are turned on (by signal RST), thus biasing inverters M1-M2 and M3-M4 at their respective switching thresholds. Thus, X and XB are biased at VM (switching threshold) of M1-M2 and Y and YB at VM of M3-M4. The pulse RST controls

switches S_{AZ} and RSTB (inverted RST) controls switches S_{REG} as shown in Figure 3(b). Pre-charge transistor M7 controlled by RSTB (inverted RST) pre-charges the NSR-SA input (IN), and node Z during reset. So, at the end of the reset, one terminal of C1 is at VDD, and the other terminal is at the VM of M1-M2. The reset pulse ends just before the WL pulse arrives. For the duration of the WL pulse, the NSR-SA is coupled to the BL by the column multiplexor, as shown in Figure 3(c). In our case, the reset pulse is 0.35n which is just a little longer than the decode time.

As soon as the WL pulse goes high the BL starts drooping (in case the bit-cell contains a zero). Also, for the duration of the WL pulse, the BL is connected to the NSR-SA by the column multiplexor. Thus, the BL droop gets coupled to the inverter M1-M2 input XB through the AC coupled input capacitor, and then ripples through C2 and M3-M4. Thus, X goes high and Y goes low. This turns on the regeneration transistor M5, further accelerating the process. Y reaches a low much faster than the BL because of inverter gain and the regenerative feedback provided by M5, thereby realizing sense amplifier behavior. Eventually M6 turns on discharging the pre-charged node Z. The final output of the NSR-SA, is the inverted version of the logic stored in the bit-cell. Thus, a delay T_{NSR-SA} from the rising edge of the WL pulse, a valid output is available. The WL pulse remains high till a valid output develops on the NSR-SA, and thus T_{NSR-SA} is set by the worst case bit-cell & NSR-SA combination. Finally the output of the NSR-SA is latched at the rising edge of the clock. It is important to notice here that the WL pulse width is equal to the T_{NSR-SA} . This happens because the NSR-SA has no strobe and it starts resolving the logic on the BL as soon as the WL turns on. In the case of reading a 1, X and Y stay close to VM thus M6 never turns on, leaving the SA output low. This timing description is available in further detail in [10]. The worse case SA Output delay sets T_{NSR-SA} , which is nothing but the WL pulse width.

This worst case is found to be near 0.71ns. Simulations on the 512-row WL decoder and driver show that T_{DEC} is 0.28ns at the typical corner. The WL driver delay is 0.04ns. However, reset happens in parallel with WL decode and driver operation.

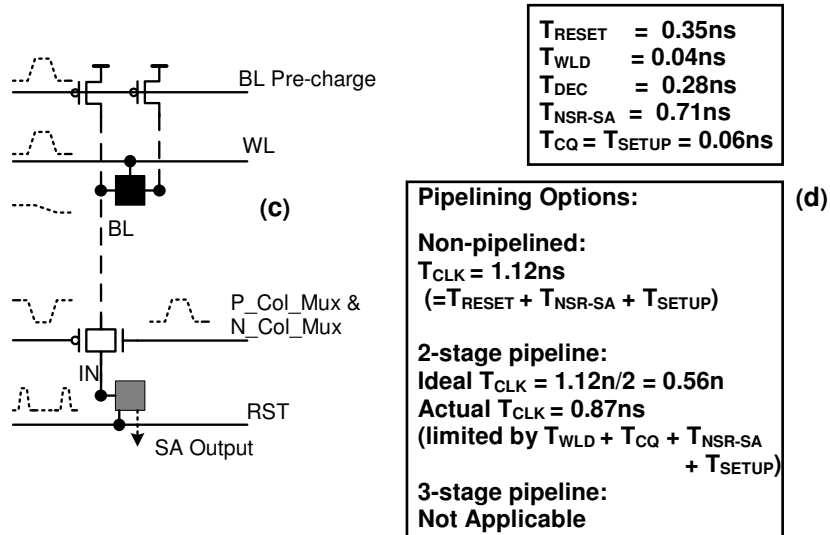
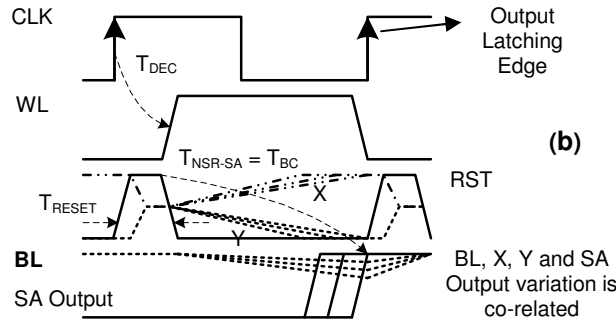
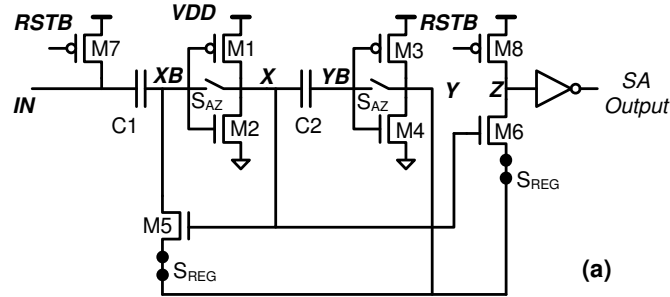


Figure 2.3: NSR-SA (a) circuit [7] and (b) timing; RSTB is inverted RST. BL signal shows variation due to variation in bit-cell. SA Output is dependent on bit-cell variation and SA variation (c) Circuits on a single column of a non-strobed SRAM with NSR-SA (d) Pipelined timing for a non-strobed SRAM with the regular read operation split into 2 cycles; the parallel operation of BL droop development and SA resolution (T_{NSR-SA}) limits cycle time. 3-stage pipeline is not possible because the non-strobed read above has only 2 distinct phases. Timing numbers are at typical corner and worst case across 100K MC simulations, 45nm technology. VDD is 1V.

Looking at the waveforms in Figure 2.3(b), the read operation of the non-strobed SRAM can be split into only two phases: WL Decode (of duration T_{DEC}) and the sum of WL driver delay and BL droop-SA resolution combination (T_{NSR-SA}). Note that unlike in strobed sensing, BL droop and sensing happen simultaneously. Thus, instead of three phases we now have only two distinct phases to partition the read operation into. But like before we still have a long unbalanced T_{NSR-SA} stage, and a short T_{DEC} stage, making pipelining inefficient. In fact now the minimum clock period for 2 stage pipeline is 0.87ns, only slightly better than the 2-stage pipeline using a strobed SA (Figure 2.2) at 0.91ns, but still much higher than the ideal value of 0.56ns ($=1.12\text{ns}/2$) in this case.

2.3.2. Non-Strobed Sensing with Truncated WL Pulse Width

Now we show how the NSR-SA can be used with modified timing to shorten the BL droop development time at the cost of SA delay, thereby improving SRAM pipelining by having balanced pipeline stages. As we mentioned before, shortening the BL droop development phase is equivalent to truncating the WL pulse width. So, we try to truncate the WL pulse width, and see the impact on NSR-SA behavior.

Figure 2.4 shows the timing waveforms of the NSR-SA with a truncated WL pulse. A smaller WL pulse means less BL droop. This means that XB, X, YB, and Y have slower slews than shown in Figure 2.3. With a long WL pulse, XB tends to go low both due to coupling from the drooping BL and due to the regeneration transistor M5. We now truncate the WL pulse such that BL goes low only till X and Y are different enough to turn on M5, but after that point further change in X, XB, Y and YB happens only because of M5 and not because of coupling from the BL. In other words, truncating WL pulse simply increases the NSR-SA delay. In this manner, we can trade-off the WL pulse width (T_{BC}) with NSR-SA delay (T_{NSR-SA}). It is very important to

notice that now we again have 3 phases in the timing waveforms: Decoder (T_{DEC}), BL droop development (T_{BC}), and the portion of the NSR-SA delay after the WL goes down ($T_{NSR-SA} - T_{BC}$).

It is also very important to notice that, now, to turn off the WL pulse we are not waiting for the NSR-SA to develop an output. Rather we turn off the WL pulse lot earlier, with the cost being higher NSR-SA delay (1.06ns as compared to 0.71ns in Figure 2.3). The key observation in truncating the WL pulse width is that the bit-cell access delay T_{NSR-SA} is dependent on the WL pulse width (T_{BC}). If the WL pulse width is large, the BL droop given to NSR-SA is high, and it evaluates very fast. If the WL pulse width is small, the BL droop given is low, and the NSR-SA evaluates slowly, but correctly. The WL pulse only needs to be wide enough such that M5 turns on sufficiently ($V_{GS} > \sim 0.2V$, although sub-threshold conduction starts regeneration even with a slightly positive V_{GS}). This critical pulse width is much smaller than the WL pulse width of Figure 2.3.

Thus we now have a mechanism to drastically reduce the WL pulse width, which is critical in increasing pipelining efficiency. In other words, the WL pulse width gives us an easy knob to trade-off the WL pulse width (T_{BC}) with NSR-SA delay (T_{NSR-SA}), which is nothing but a mechanism of balancing stage delay. Note that the RST pulse which is shown to come soon after the WL ends (in Figure 2.3) is not shown in Figure 2.4. In the timing of Figure 2.3, the RST pulse could come after the WL pulse ended because the slowest output was ready by the time WL pulse ended. However, in the truncated WL case, we need the RST pulse to be delayed so that the NSR-SA can take its time and slowly develop the correct output after the WL pulse ends. We will discuss a mechanism to solve the reset signal issue in the next section. But first we

would complete the picture by using the truncated WL concept in implementing a pipelined SRAM using non-strobed sensing.

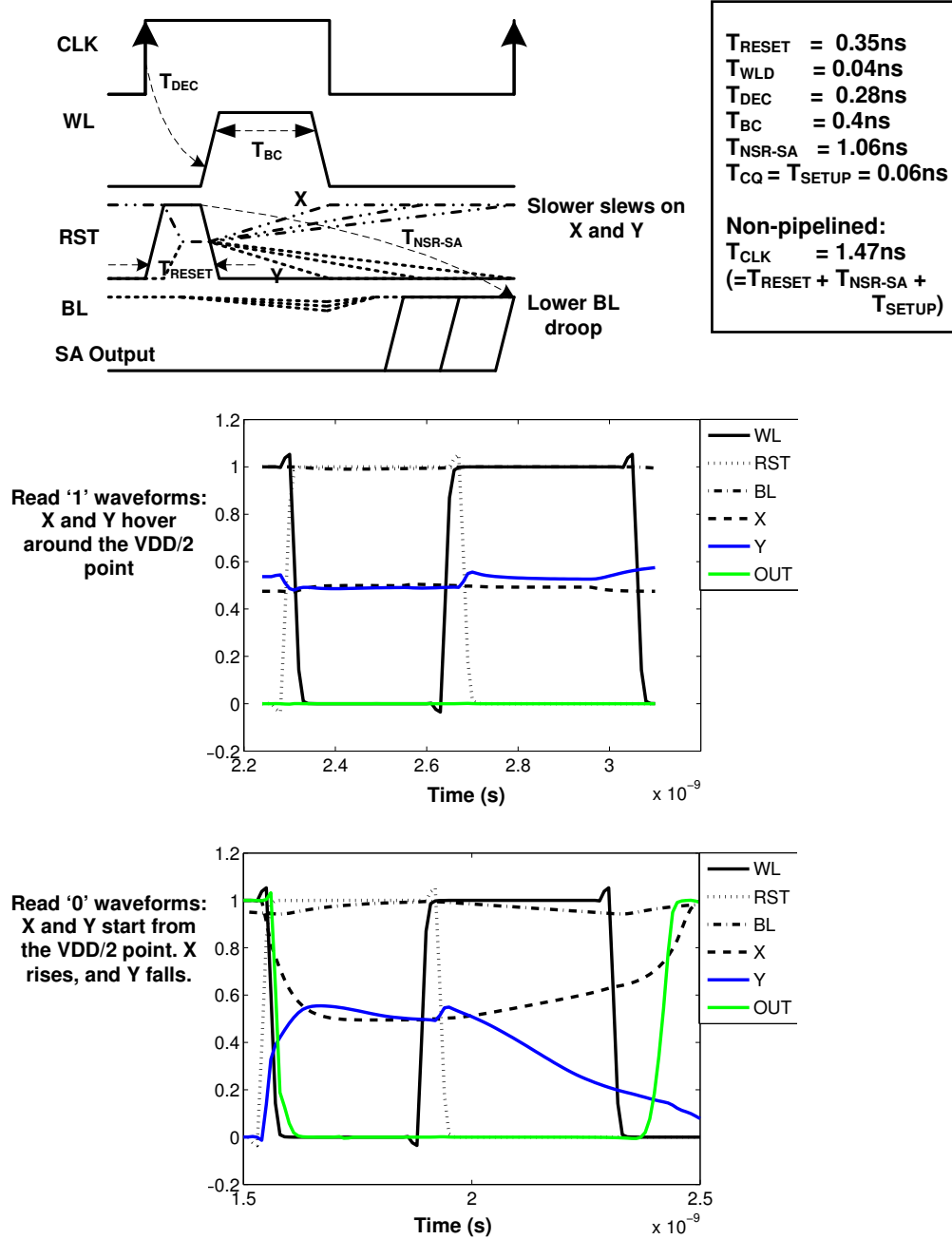


Figure 2.4: NSR-SA timing and waveforms with truncated WL for a single read operation; Shorter WL pulse causes lesser BL droop, and thus makes X, Y & SA Output slower than in Figure 2.3(b). Timing numbers are at typical corner and worst case across 100K MC simulations, 45nm technology. VDD is 1V.

2.4. Non-strobed pipelined sensing scheme: Circuit implementation and Results

Looking at the waveforms in Figure 2.4, the read operation of the non-strobed non-pipelined SRAM with truncated WL can be split into 3 phases: WL Decode (T_{DEC}), the truncated WL pulse width (or in other words the truncated BL droop development phase, T_{BC}), and the portion of the NSR-SA resolution time after the WL has gone low ($T_{NSR-SA} - T_{BC}$). By controlling the T_{BC} vs. T_{NSR-SA} trade-off, we can set the timing such that first two phases together have duration ($T_{RESET} + T_{BC}$) close to the last phase ($T_{NSR-SA} - T_{BC}$). We pipeline the non-strobed SRAM into these two balanced phases, thereby achieving efficient SRAM pipelining. Also, it is very important to notice that unless we pipeline the non-strobed SRAM, truncating the WL has a detrimental impact on the clock period. Figure 2.3, with long WL pulse had clock period of 1.12ns for the non-pipelined case, while Figure 2.4, with truncated WL pulse has clock period of 1.47ns for the non-pipelined case.

Figure 2.5 shows the simulated waveforms for the NSR-SA with truncated WL and normal WL. An extreme case out of the 100K montecarlo runs (at typical corner) is shown. The value of the truncated WL pulse width in Figure 2.5a is chosen through an iterative process of montecarlo simulations such that $(T_{RESET} + T_{BC}) = (T_{NSR-SA} - T_{BC})$. Through this iterative process we arrive at WL pulse width of 0.4ns (100K sims, typical corner). T_{RESET} remains same as that for the un-truncated WL case at 0.35ns. It can be seen from the waveforms that the NSR-SA generates output with a higher delay because of the truncated WL as compared to waveforms in Figure 2.5b.

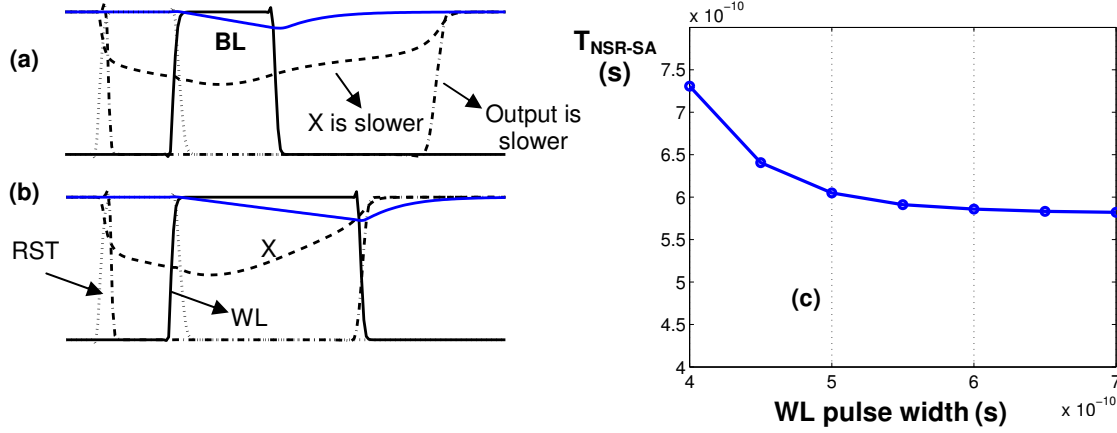


Figure 2.5: Shows the timing waveforms for (a) truncated and (b) normal WL cases. As WL pulse width is reduced, internal waveforms like X and Y becomes slower, and as a result the SA output arrives later. Same is shown in the graph (c) where as WL increases, SA delay reduces, but then saturates as WL pulse width exceed the SA delay itself.

The waveforms in Figure 2.6(a) show the NSPS timing scheme. We pipeline the non-strobed SRAM read operation into two cycles. The first cycle includes WL decode (T_{DEC}), and the truncated WL pulse (or in other words, the truncated BL droop development, T_{BC}). The second cycle is the portion of the NSR-SA resolution time after the WL has gone low ($T_{NSR-SA} - T_{BC}$). The column mux (Figure 2.6b) at the boundary of bit-cell and SA acts as a dynamic transmission gate latch. Further, by controlling WL pulse width, both stages can be balanced. Figure 2.6(a) shows the SRAM with two back to back read operations. After the first WL pulse goes down, the next cycle (and read access) starts immediately. During this cycle, the next address is accessed, and in parallel, the NSR-SA is evaluating the previous read. BL pre-charge occurs in the beginning of each cycle in parallel with the WL decode, and is not in the critical timing path. In our case the reset is in the critical path. Figure 2.6(b) shows the circuits associated with a single column of the SRAM. When the WL goes high, the NSR-SA is coupled to the BL. When the WL goes down, the NSR-SA is decoupled from the BL, but its input remains at the same level. We optimize T_{BC} such that first two phases together have duration ($T_{RESET} + T_{BC}$) close to the last phase ($T_{NSR-SA} - T_{BC}$). Thus the clock period of the 2-stage pipeline is set at 0.75ns ($=T_{RESET} +$

T_{BC}). This is lowest clock period of all 2-stage pipeline options mentioned in this chapter, and is equal to the clock period of the 3-stage pipeline using strobed sensing (Figure 2.2). Thus, by using NSPS, we make the SRAM clock cycle faster by 18% as compared to the 2-stage pipelined strobed SRAM, and 1.6X faster than non-pipelined strobed SRAM. Note that 3-stage pipeline has lower CPI than a 2-stage pipeline (1.26 vs. 1.13), thus a 2-stage non-strobed SRAM would still be faster than a 3-stage strobed SRAM. A 3-stage NSPS would divide the read operation into decode, truncated BL droop development and NSR-SA delay after the WL goes down. Here the optimal WL pulse width can be found through an iterative process that sets $T_{BC} = T_{NSR-SA} - T_{BC}$. However, as mentioned before 3-stage pipelines increase area because of addition of a flip-flop in each WL driver, and also have lower CPI. Hence, we focus on 2-stage pipelines. Figure 2.7 summarizes the various pipelined and non-pipelined options.

As mentioned briefly in the previous section, a circuit issue in implementing NSPS is the timing of the reset pulse. Let us take the example of the back to back read operations in Figure 2.6(b). During cycle 2, the NSR-SA is resolving the BL signal from the read access of cycle 1. However, during cycle 2, the NSR-SA also needs to be reset and connected to the BL for sensing the read in the current read access of cycle 2. This is a structural hazard in the SRAM pipeline. To resolve this structural hazard, we use a Sense Amplifier Alternation (SAA) scheme. The scheme involves having two NSR-SAs per column and alternating between them during consecutive read accesses. One NSR-SA is connected to BL, and the other to BLB. This way, both NSR-SAs get an entire cycle to develop the output. In every cycle one NSR-SA is in reset and BL droop development mode, and the other is in signal resolving mode. Adding an extra sense amplifier is a small area overhead in the overall SRAM context, especially because most

commercial SRAMs have a heavy degree of column multiplexing. For example, in an SRAM with 8to1 column multiplexing, we would only need two NSR-SAs per 8 columns.

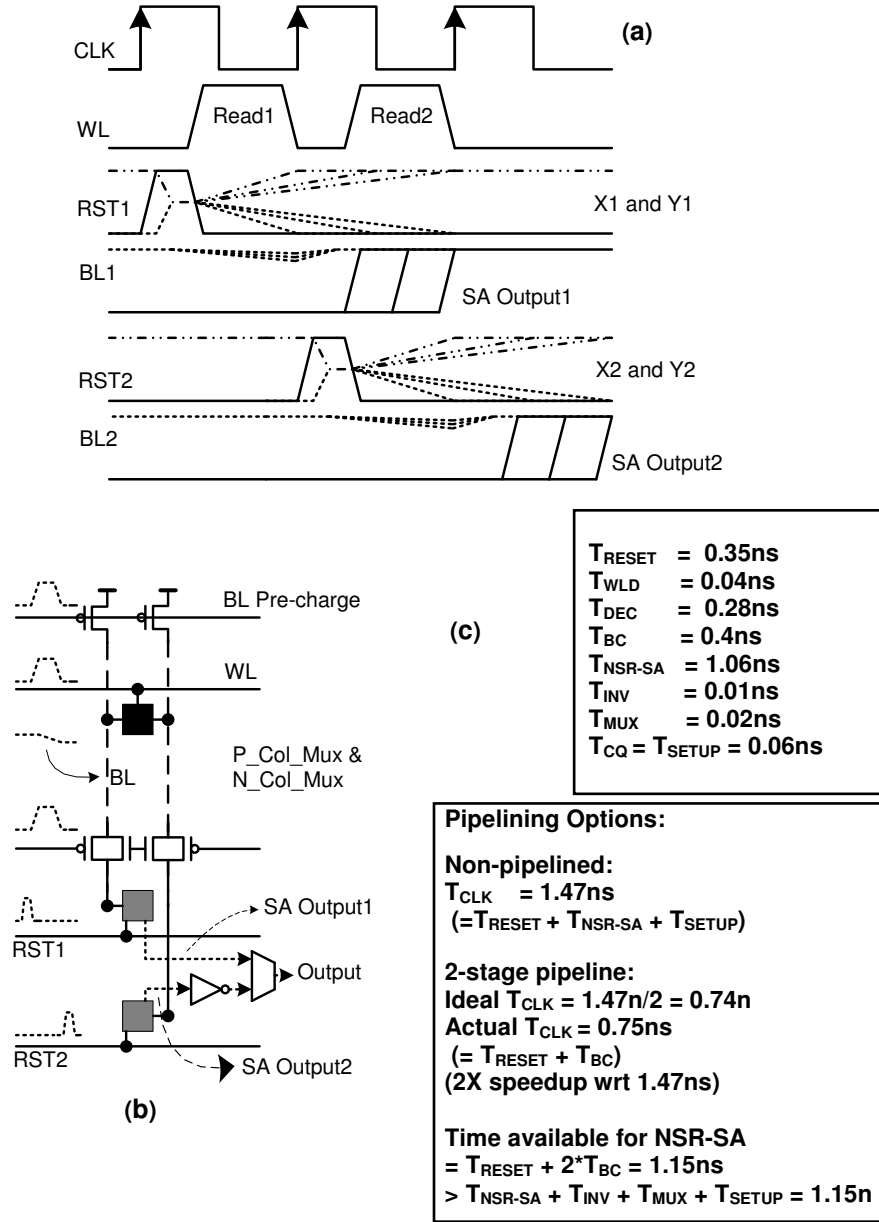


Figure 2.6: (a) NSPS timing for 2 back to back read operations; two sets of signals for the NSR-SA correspond to two NSR-SA that are used in alternate cycles to resolve a structural hazard in the SRAM pipeline (b) Circuits in a single column of the NSPS SRAM (c) Pipelined timing for NSPS SRAM with the regular read operation split into 2 cycles; The reset (T_{RESET}) and BL droop development (T_{BC}) are balanced with the *portion* of sensing time *after* WL goes low ($T_{\text{NSR-SA}} - T_{\text{BC}}$). Timing numbers are for a typical corner and worst case across 100K MC simulations, 45nm technology. VDD is 1V.

In the design of the NSPS scheme, it is important to ensure that once the WL goes low and the NSR-SA gets decoupled from the BL, the node IN remains at the drooped level where it was when the WL went low. For this, we need to ensure that the charge injection and clock feed-through noise injected into the node IN by the column multiplexor is low. To lower the noise, we use transmission gate based column multiplexors with small sizes.

In all simulations for this work, typical corner is used and 100K montecarlo sims are run for measuring bit-cell and SA variations. Appropriate parasitics are used on the BL to generate the delay numbers. For finding T_{DEC} a 512 row static CMOS decoder is built, and the T_{BC} , T_{NSR-SA} numbers are found by using a single column SRAM model like in Figure 2.6b. Since the NSPS scheme only changes circuits in this column, we estimate the power overhead in an SRAM using these simulation results. The energy per read operation of a non-pipelined SRAM using strobed sensing is 3.6pJ for 32-bitwidth, 8 to 1 column mux-ing, and 512 rows, counting the energy in BLs and SAs. This only includes energy from the column (as in Figure 2.3c) being simulated, and then multiplied by the number of selected and unselected columns. For NSPS the energy per operation is 2.6pJ. NSPS has energy overhead due to use of the non-strobed SA which burns static power during its reset phase and also while evaluating a '1'. However, lower droop on the BLs due to the truncated WL results in massive energy saving that compensates for the use of the more power hungry NSR-SA as compared to a strobed SA. Overall, the NSPS scheme has 28% lower energy per read as compared to the non-pipelined strobed SRAM. Note that this is the most pessimistic energy comparison possible, because all other pipelined SRAM versions will have some overhead due to pipelining. In [10], NSR-SA area and conventional strobed SA area was reported $19\mu m^2$ and $12\mu m^2$ respectively. Since we use two NSR-SAs, we can estimate our SA area to be about 2.5X the NSR-SA. Taking bit-cell area to be $0.25\mu m^2$ and a 512x256, 32-bit

output SRAM with 8 to 1 column muxing, and 75% array efficiency, on the SRAM level, there is a 2.9% overhead of using NSPS because of the bigger SAs. This is a small area penalty, and is something that's expected because most area in an SRAM is consumed by bit-cells.

STROBED SENSING

$$T_{DEC} = 0.28ns \quad T_{BC} = 0.69ns \quad T_{SA} = 0.2ns \quad T_{CQ} = 0.06ns$$

Pipelining Options:

Non-pipelined:

$$T_{CLK} = 1.23ns$$

$$(= T_{DEC} + T_{BC} + T_{SA} + T_{SETUP})$$

2-stage pipeline:

$$\text{Ideal } T_{CLK} = 1.23n/2 = 0.62n$$

$$\text{Actual } T_{CLK} = 0.91ns$$

$$(= T_{DEC} + T_{BC})$$

$$(1.35X \text{ speedup wrt } 1.23ns)$$

3-stage pipeline:

$$\text{Ideal } T_{CLK} = 1.23n/3 = 0.41n$$

$$\text{Actual } T_{CLK} = 0.75ns$$

$$(= T_{BC} + T_{CQ})$$

$$(1.64X \text{ speedup wrt } 1.23ns)$$

For 512 row,
8 to 1 col mux , 32b output:
Energy per read = 3.6pJ

NON-STROBED PIPELINED SENSING (NSPS)

$$T_{RESET} = 0.35ns$$

$$T_{WLD} = 0.04ns$$

$$T_{DEC} = 0.28ns$$

$$T_{BC} = 0.4ns$$

$$T_{NSR-SA} = 1.06ns$$

$$T_{INV} = 0.01ns$$

$$T_{MUX} = 0.02ns$$

$$T_{CQ} = T_{SETUP} = 0.06ns$$

For 512 row,
8 to 1 col mux , 32b
output:
Energy per read = 2.8pJ

Pipelining Options:

Non-pipelined:

$$T_{CLK} = 1.47ns$$

$$(= T_{RESET} + T_{NSR-SA} + T_{SETUP})$$

2-stage pipeline:

$$\text{Ideal } T_{CLK} = 1.47n/2 = 0.74n$$

$$\text{Actual } T_{CLK} = 0.75ns$$

$$(= T_{RESET} + T_{BC})$$

$$(2X \text{ speedup wrt } 1.47ns)$$

Time available for NSR-SA

$$= T_{RESET} + 2 \cdot T_{BC} = 1.15ns$$

$$> T_{NSR-SA} + T_{INV} + T_{MUX} + T_{SETUP} = 1.15n$$

Figure 2.7: Non-strobed pipelined sensing with truncated WL gives lowest clock period of 0.75ns across all options. This is 1.6X faster than a strobed non-pipelined SRAM. It is 1.2X faster than the 2-stage non-strobed SRAM with long WL. It is as fast as the 3-stage pipelined strobed SRAM, but has lower CPI (1.13) than a 3-stage pipeline (1.26). VDD is 1V. Energy numbers are estimated for 512 row, 32b output 8 to 1 col mux SRAM.

2.5. Conclusion

In this chapter we presented methods to reduce the cycle time of instruction memory read operation by pipelining. SRAM pipelining is tough to implement due to the long unbalanced BL droop development stage. BL droop development in a conventional strobed SA cannot be split into two cycles, thus it limits the efficiency of pipelining. We explore the use of non-strobed sensing to solve this problem. In our technique called Non-Strobed Pipelined Sensing (NSPS), we propose the use of a truncated WL pulse, which increases SA delay, but helps by shortening the BL droop development period. This enables us to split the SRAM read operation into two equal pipeline stages. NSPS helps increase performance by 1.6X as compared to a non-pipelined strobed SRAM, while also decreasing energy per read by 28%. As compared to a 2-stage pipelined strobed SRAM, NSPS is 18% faster. The area penalty is less than 3%.

Chapter 3: A Fast Low Power Non-Strobed Differential Sense Amplifier and Sensing Scheme for Asynchronous SRAMs

To achieve high density, SRAMs use bit-cells with small transistor sizes. As a result the bit-cells dominate SRAM read delay. Local variation in nano-scale technologies compounds the problem as the near minimum sized transistors of bit-cells are worst hit by such variation. The conventional solution is to estimate the variation and to include enough margins in SRAM timing. This chapter describes a novel non-strobed differential sense amplifier and a sensing scheme where the delay of the SRAM varies with the bit-cell being read. As a result the SRAM is fast most of the times, but is slow while reading the few slow bit-cells. The scheme improves average SRAM delay by 17%. The sensing scheme is applicable for asynchronous SRAMs, but the differential sense amplifier circuit can be used in synchronous SRAMs as well.

3.1. Introduction

SRAMs are a critical component in most modern digital systems because they are a fast, low power, logic technology compatible form of storage. To lower area and hence cost per chip, significant amount of effort is put into lowering bit-cell area. At the same time, technology scaling lowers area by half every two years further reducing cost. However, as technology scales, local variation in transistor parameters increases. Further, small transistors like those used in SRAM bit-cells are worst affected by this variation [4]. As a result, SRAM delay is limited by the weakest bit-cell. Variation in bit-cell read current is near-gaussian, but with multi-million bit-cells on a die, the tail of the distribution can be more than 5-6 sigma away from the mean. Thus the worst case delay of a bit-cell is often 2-3 higher than the mean delay.

In this chapter, we describe a sensing scheme for asynchronous SRAMs which reacts to the variation in each bit-cell rather than using a worst case timing approach. As a result the SRAM is fast most of the times, but is slow while reading the few slow bit-cells that lie at the tail of the distribution. Correct data is made available along with a data valid signal as soon as correct data is available rather than a worst case approach [13]. Thus over a large number of reads, the memory has lower average delay. Conventional (strobed) sense amplifier topologies [6] cannot realize this adaptive behavior because they use a fixed timing pulse to enable the sense amplifier based on the studied worst case variation. We propose two sense amplifier circuits that help realize the required behavior of our sensing scheme. We show circuit and architectural techniques that help achieve this adaptive behavior and make the scheme usable in asynchronous digital systems. The differential sense amplifier itself can be used in synchronous SRAMs when performance and robustness are critical parameters, and power can be traded-off. Drawing from our designs and analysis, this chapter makes the following key contributions:

- Shows that conventional (strobed) sensing can only be designed with a worst case timing scheme and thus cannot be used to design an (asynchronous or synchronous) SRAM whose delay depends on the particular bit-cell being read.
- Demonstrates a sensing scheme using a novel differential non-strobed sense amplifier to enable local-variation-adaptive SRAM delay. The technique improves asynchronous SRAM average read delay by up to 17%.

This chapter is organized in the following manner. Section 3.2 analyses conventional (strobed) and non-strobed sensing schemes. Section 3.3 describes the novel non-strobed differential sense amplifier. Section 3.5 describes the application of the proposed SA to an

asynchronous sensing scheme that helps lower average case SRAM read delay. Section 3.6 concludes the chapter.

3.2. Timing in strobed and non-strobed sensing schemes

In this section we analyze the operation of conventional (strobed) and non-strobed sensing schemes. We would refer to conventional sensing as strobed sensing from here on. A strobed sensing scheme works as shown in Figure 3.1. SRAM sub-blocks use timing signals generated by a timing block using the system clock and delay elements. A delay T_{DEC} after the rising clock edge the SRAM timing block generates the word-line (WL) pulse. This delay is set by the row decoder delay or the bit-line (BL) pre-charge delay, whichever is higher. Then, a delay T_{BC} after the WL enable rising edge, the sense amplifier enable (SAE) pulse arrives, as shown in Figure 3.1. T_{BC} is set such that the weakest bit-cell has enough time to pull the BL low enough that the voltage difference between BL and BLB is greater than the sense amplifier (SA) offset. The SA has a finite resolution time, and this sets the width of the SAE pulse, T_{SA} . It is these set delays, which makes the strobed sensing scheme invariant to bit-cell strength. Even if a stronger bit-cell is being read, the SRAM read delay will be fixed at $T_{DEC} + T_{BC} + T_{SA}$. Note that for asynchronous systems the clock will be replaced by a synchronization pulse given by the processor to the SRAM.

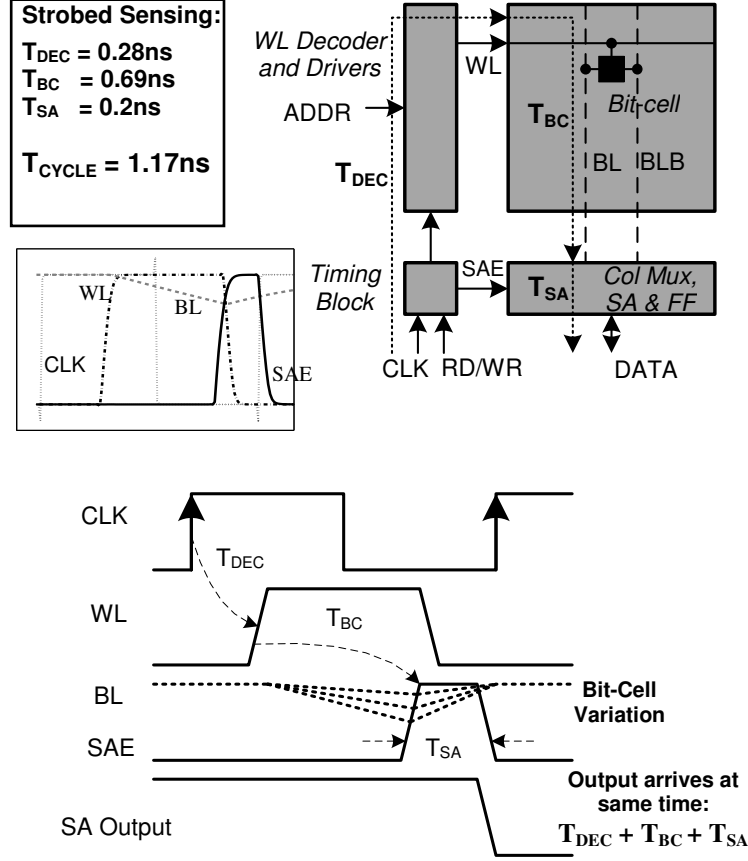
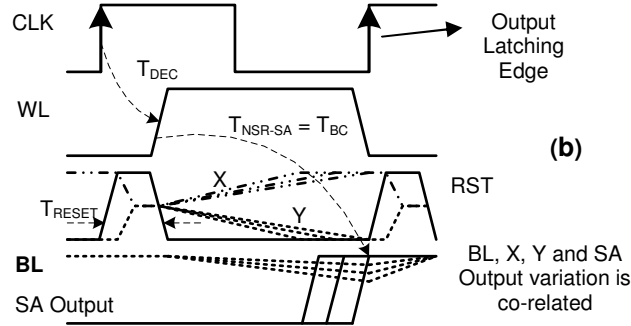
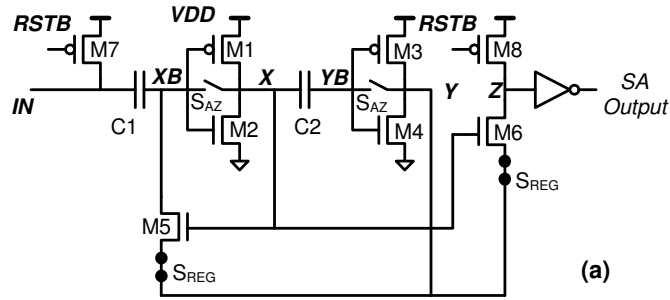


Figure 3.1: Conventional strobed SRAM macro with timing arches (dotted). Timing numbers for typical corner and worst case across 100K MC sims. SRAM is 512 rows, 16-bit wide, 8 to 1 column mux-ing. VDD is 1V.

The timing of a non-strobed sensing scheme is significantly different. In the next section we describe a novel differential non-strobed sense amplifier. Our SA is based on the single ended non-strobed SA called NSR-SA from [10]. The NSR-SA timing is shown in Figure 3.2. The NSR-SA needs a resetting pulse (of width T_{RESET}) just before the WL turns on. This resetting pulse biases the two inverters M1-M2 and M3-M4 at their respective switching thresholds by turning on switches S_{AZ} . The reset pulse ends almost as the WL pulse arrives. As soon as the WL goes high the bit-line starts drooping in case the bit-cell contains a zero. This droop gets coupled to the inverter M1-M2 input through the input capacitor, and then ripples to the output of the sense amplifier. The output of the sense amplifier reaches a low much faster than the bit-line

droop because of inverter gain and the regenerative feedback provided by M5, thereby realizing sense amplifier behavior. Thus, $T_{\text{NSR-SA}}$ after the WL pulse, a valid output is available. $T_{\text{NSR-SA}}$ depends heavily on the bit-cell being read, and to a small extent on the strength of the transistors in the sense amplifier. The WL pulse remains high till a valid output develops on the NSR-SA [10], and is thus set by the worst case bit-cell. In the case of reading a '1', X and Y stay close to V_M thus M6 never turns on, leaving the SA output low. To make the NSR-SA robust to noise while reading a '1', the switches S_{AZ} are designed such that they inject noise that lowers X and raised Y by default. This adversely impacts the NSR-SA delay while reading a '0'. There are two important differences from the strobed case. First, there is no SAE pulse required. Second, because $T_{\text{NSR-SA}}$ depends on the bit-cell strength, a stronger bit-cell bring read would result in a lower $T_{\text{NSR-SA}}$. Finally the output of the NSR-SA is latched at the next rising edge of the clock. It is important to notice here that the clock period, and thus the time when the clock edge latches the NSR-SA output must be after the slowest $T_{\text{NSR-SA}}$. Thus, when used in the manner showed in Figure 3.2, NSR-SA delay is also limited by the worst case bit-cell. In the next two sections we show how the our differential non-strobed SA can be used with a modified timing to implement an asynchronous SRAM with read delay dependent on the bit-cell being read. Again, for asynchronous systems the clock will be replaced by a synchronization pulse given by the processor to the SRAM. Latching at the clock would be replaced by use of the SRAM data at a time specified by the SRAM self timed mechanism (e.g. a pulse that is generated by the SRAM at the end of the SAE).



T_{RESET}	$= 0.35\text{ns}$
$T_{\text{NSR-SA}}$	$= 0.71\text{ns}$
T_{CYCLE}	$= 1.06\text{ns}$

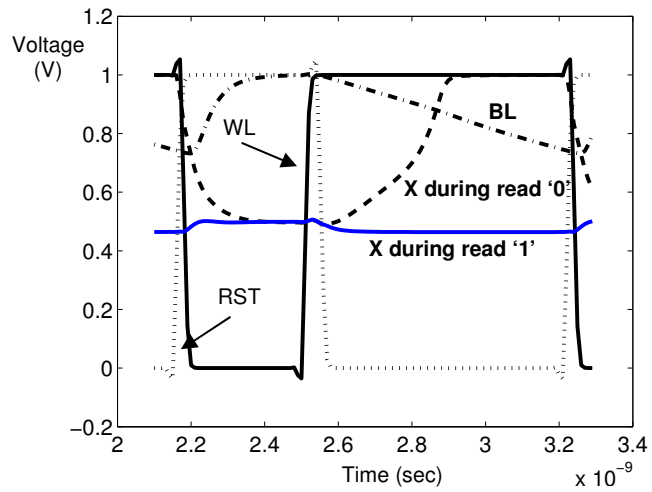
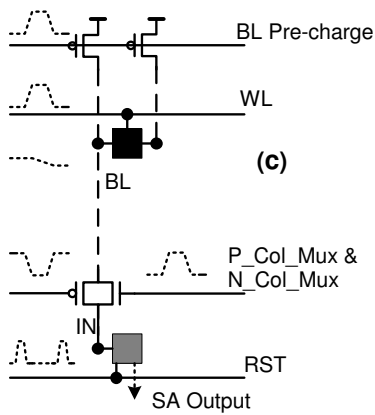


Figure 3.2: NSR-SA (a) circuit [7] and (b) timing; RSTB is inverted RST. BL signal shows variation due to variation in bit-cell. SA Output is dependent on bit-cell variation and SA variation (c) Circuits on a single column of a non-strobed SRAM with NSR-SA Timing numbers are at typical corner and worst case across 100K MC simulations, 45nm technology. SRAM is 512 rows, 16-bit wide, 8 to 1 column mux-ing. VDD is 1V.

3.3. Differential Non-Strobed Sense Amplifier

The NSR-SA is a single ended SA. In this section we will modify the NSR-SA to obtain a differential non-strobed SA. This section describes the DNSA qualitatively. We will present power and performance numbers in the next section. Figure 3.3 shows the circuit diagram and waveforms of the differential non-strobed SA (DNSA). DNSA consists of two NSR-SAs at its core: one each for BL and BLB. The two NSR-SAs are then cross coupled by the capacitors C_{CROSS1} and C_{CROSS2} . The operation can be understood by first ignoring the presence of the cross-coupling capacitors. Without the cross-coupling capacitors, the DNSA is nothing but two NSR-SAs. Thus, when reading a '0' in a bit-cell, the NSR-SA connected to BL will see coupling into its internal nodes. X will go high, and Y will go low. On the other hand, there would be no voltage change on BLB, thus no voltage will couple into the NSR-SA connected to BLB. Thus X1 and Y1 will remain close to $VDD/2$. In other words, X1 and Y1 (and XB1 and YB1) are not driven to any voltage level. This can make then very sensitive to noise. This happens during the normal operation of the NSR-SA while reading a '1'. To avoid this un-driven scenario, we introduce cross-coupling in the DNSA. Now, when any side reads a '0', it provides an opposite feedback to the other side. This means that now, while reading a '1', nodes inside the DNSA are driver, and not left floating. This makes them more robust to noise. The additional feedback acts as a second feedback in addition to the regular feedback provided by M5. Also, we don't need to rely on charge injection to make the read '1' robust, thus we don't adversely impact the read '0' delay. This makes the DNSA faster than NSR-SA at the same noise level or more robust to noise at the same delay. The penalty is the extra area consumed by the additional capacitor, however this is going to be less than 5% in the context of an entire SRAM. We have set the cross-coupling

capacitor to half the size of the other series capacitors. The cross-coupling capacitors adversely impact T_{RESET} . This metric decides their sizing.

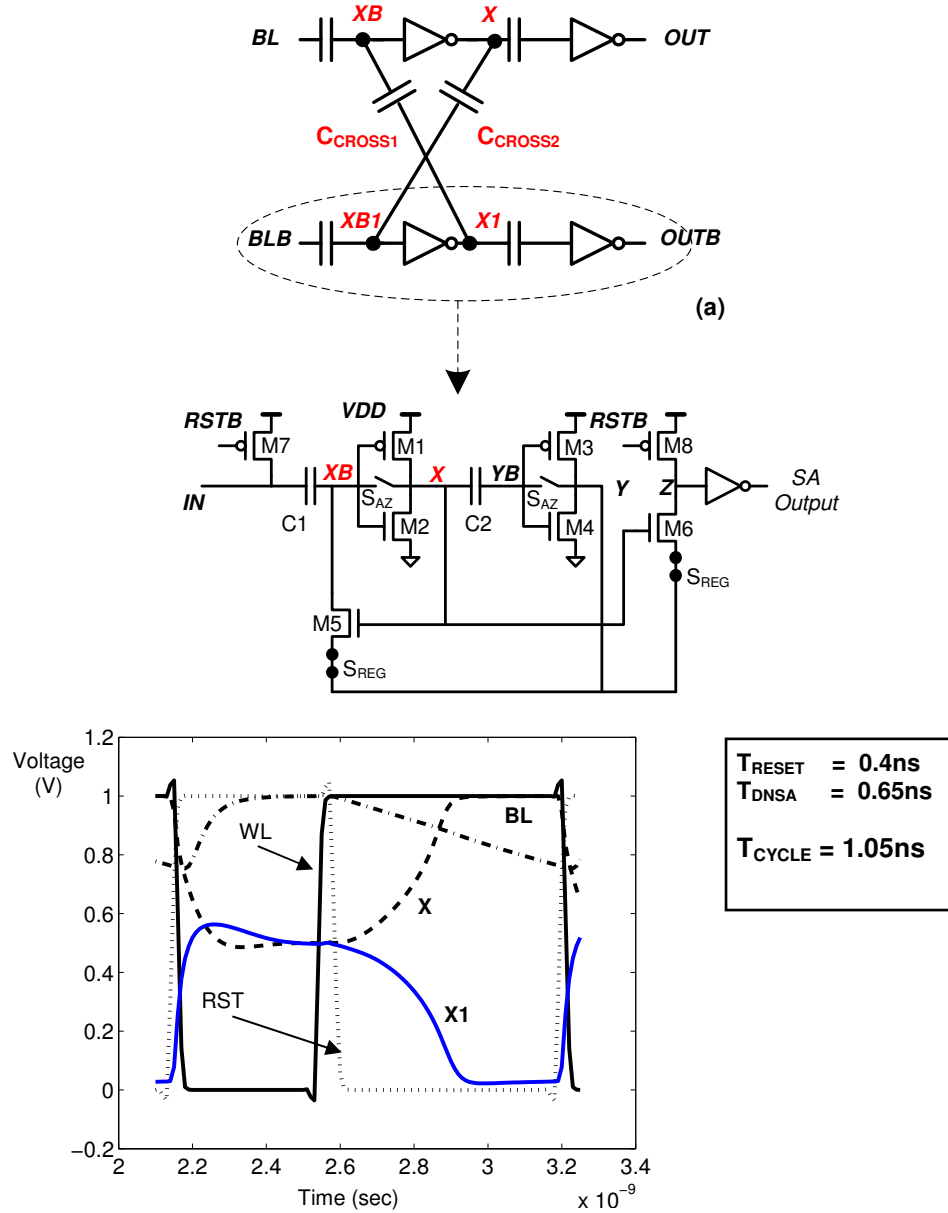


Figure 3.3: DNSA (a) circuit uses two NSR-SAs at its core, which are coupled by two capacitors (b) waveforms. Timing numbers are at typical corner and worst case across 100K MC simulations, 45nm technology. SRAM is 512 rows, 16-bit wide, 8 to 1 column mux-ing. VDD is 1V.

3.3.1. BL Noise Analysis

In this section, we will analyze the impact of common mode BL noise on DNSA and NSR-SA. As mentioned before, DNSA is differential in nature, and thus has the capability to reject most of the common mode noise. Noise on the BL impacts the SA if the noise occurs during the time the SA is resolving. As shown in Figure 3.4a, if the data being read is a '0', positive noise on the BL will cause the single ended NSR-SA to have higher delay because positive noise effectively reduces the differential that the NSR-SA is seeing. The positive noise causes X and Y nodes to slow down, and in extreme cases may even cause them to change direction, and read a '1'. Slow X and Y instead lead to a slower output. As shown in Figure 3.4b, if the data being read is a '1', negative noise on the BL will cause the single ended NSR-SA to mistake the '1' for a '0' which is a functional failure. Noise on BL causes X and Y to have amplified noise which can be so extreme that it changes the direction X and Y were going, making the NSR-SA to read a '0'. The chance of this happening depends on the magnitude of the noise, and also where in the local variation spectrum the NSR-SA is sitting. Because NSR-SA is single ended, common mode noise has a significant impact on its behavior. The table in Figure 3.4e shows the $T_{\text{NSR-SA}}$ of the NSR-SA as a function of positive noise at the nominal PTV (no local variation). A 50mv noise increases the NSR-SA $T_{\text{NSR-SA}}$ by almost 1.5X, which can easily lead to functional failures if enough margins are not introduced in the clocking strategy.

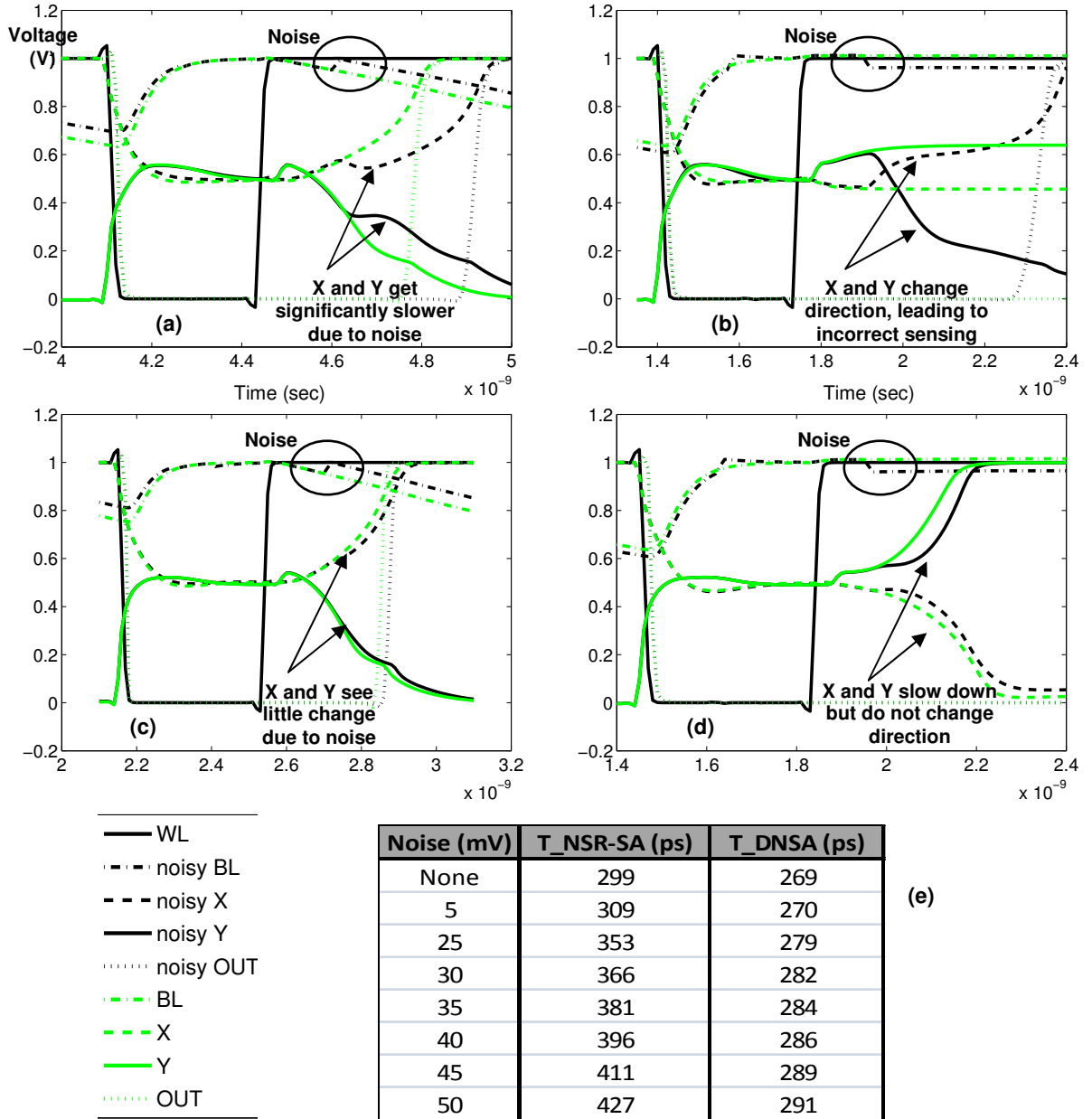


Figure 3.4: Impact of noise on NSR-SA and DNSA. (a) 50mV positive noise on BL for NSR-SA causes significant increase in delay (b) 50mV negative noise on BL causes NSR-SA to sense incorrectly (c) 50mV positive noise on BL for DNSA causes little change in delay (d) 50mV negative noise on BL delays X and Y signals but they retain their relative magnitudes, and sensing is correct (e) shows the sensing delays of NSR-SA and DNSA as a function of positive noise on BL while reading a '0'. NSR-SA shows much more sensitivity to noise. (a)-(e) are at the nominal corner, and no local variation.

The DNSA has a differential topology, and is thus capable of rejecting most of the common mode noise. A positive noise on BL and BLB while the DNSA is reading '0' on the BL causes the internal node X (in Figure 3.3a) to see a sudden dip at the moment of the noise, but the

feedback from XB1 though C_{CROSS2} causes a rise at the same time. The noise related dip, and rise almost cancel each other out, and X remains almost unchanged as seen in Figure 3.4c. Similarly, while reading a '1' on the BL, a negative noise on BL and BLB is almost fully cancelled internally, and X and Y nodes do not change direction, as seen in Figure 3.4d. It must be noted that a higher positive noise on the BL may cause the DNSA also to resolve incorrectly. However, for the same noise and at the same process corner, DNSA has more noise immunity than the NSR-SA, which is also obvious because DNSA is differential. In Figures 3.4a-d, the noise magnitude is 50mV. The table in Figure 3.4e shows that while NSR-SA T_{NSR-SA} changes by about 1.5X on application of 50mV noise, the DNSA T_{DNSA} changes by less than 1.1X.

3.4. Asynchronous SRAM with DNSA

In this section we describe an alternative to worst case timing paradigm for asynchronous SRAMs. As we described in section 3.2, by using a strobed SA or a single-ended non-strobed SA, we have to assume worst case operation. Even if the bit-cells being read in a given operation are fast, we are forced to wait for an amount of time dictated by timing pulses within the SRAM. The strobed SA is limited because it needs a timing pulse to fire, and that timing pulse has to come at a time enough for the slowest bit-cell to create enough differential. The single ended NSR-SA is limited because while reading a '1' there is no way for us to determine if the bit-cell being read is actually a '1' or a weak '0' bit-cell. We have to wait for the studied worst case time for reading a '0'.

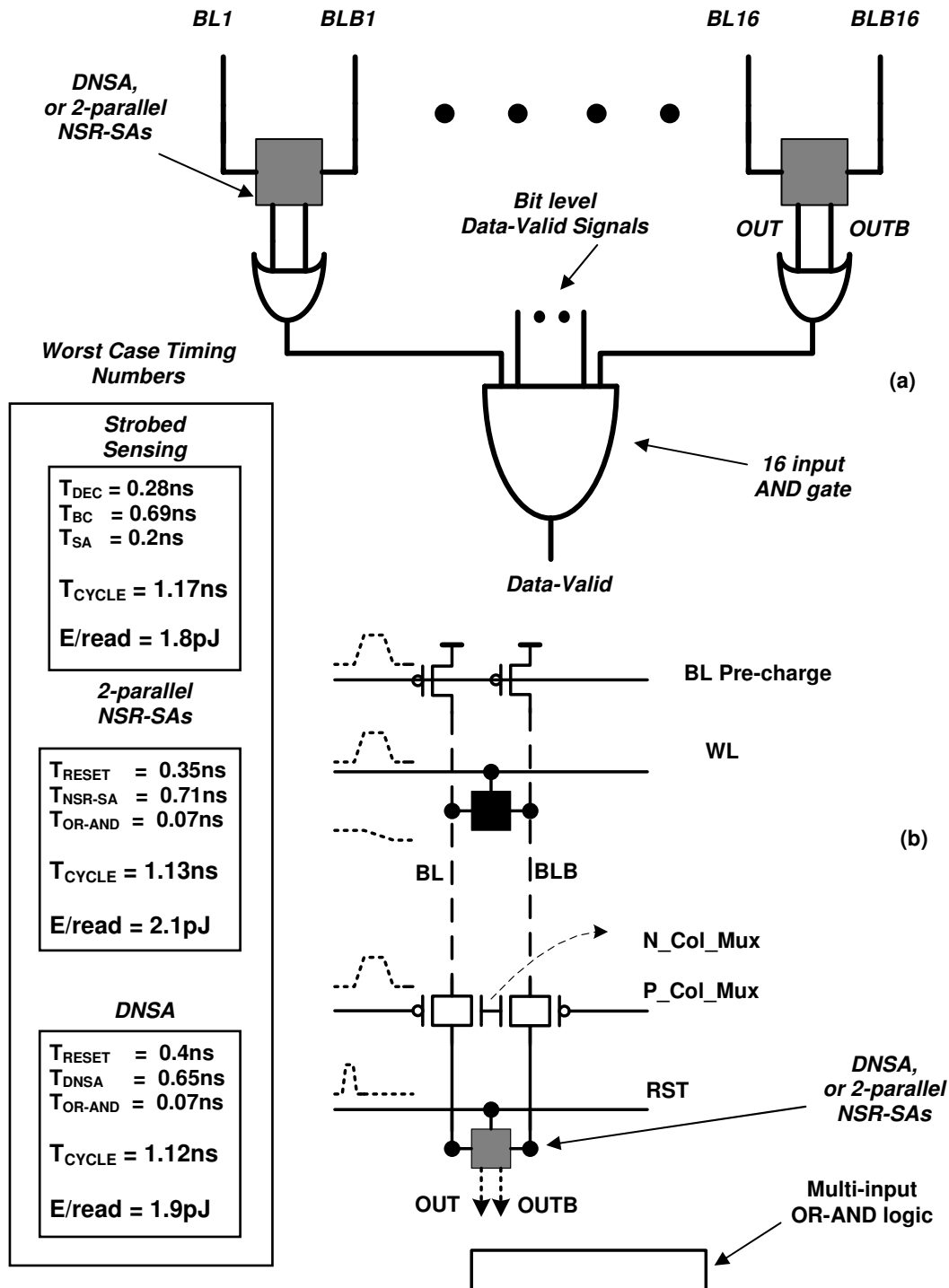


Figure 3.5: Block diagram showing DNSA being used to generate a Data-Valid signal in a 16-bit wide SRAM. The block diagram helps implement an asynchronous SRAM sensing scheme wherein the read delay of the SRAM is dependent on the bit-cell being read. SRAM is 512 rows, 16-bit wide, 8 to 1 column mux-ing. Typical corner and worst case across 100K MC simulations, 45nm technology. VDD is 1V. Energy numbers are estimated for 512 row, 16b output 8 to 1 col mux SRAM.

What we want is a SA that after correctly sensing tells the processor that it is done reading, and that the data on the data bus is valid. Moreover, this data-valid signal must arrive fast if the bit-cell is fast, and slow if the bit-cell is slow. That would ensure that our SRAM delay is not limited by the worst case. We propose two ways to achieve this. First, we can use two NSR-SAs. One connected to the BL, the other to BLB. That way, whichever side reads a ‘0’ will switch, and by OR-ing the two outputs, we can get a bit level data-valid signal. The other option is to use the DNSA, which works in a similar fashion. The side that is connected to ‘0’ switches, and pushes the other side not to switch. The sensing scheme along with the DNSA is shown in Figure 3.5. The DNSA has the feature that one of its differential outputs switch while reading. By OR-ing the outputs, we can generate a data-valid signal that indicates that the DNSA is done, and valid data is available. By AND-ing the data-valid signal of all the DNSAs, we can generate an SRAM level data-valid signal. This signal can then act as a synchronization signal to the rest of the system.

As shown in the worst case timing numbers in Figures 3.5, DNSA has a larger T_{RESET} but a shorter T_{DNSA} as compared to $T_{\text{NSR-SA}}$. The setup shown in Figure 3.5b was used in the simulations. The setup includes a single column along with dummy bit-cells, precharge, column mux, and SA. The overall read access delay is very similar for both. However, a shorter T_{DNSA} results in lower BL droop. This directly translates to lower energy per read operation in both selected and unselected columns. The DNSA itself consumes slightly more energy per read than the 2-parallel NSR-SAs. However, the energy saved in the BLs is much larger, resulting in 8.5% energy saving in the BLs and SA for a 16-bit 8 to 1 column muxed SRAM while considering energy in BLs and SAs. The results are for 45nm technology, nominal PTV, and 100k montecarlo runs. The other advantage of the DNSA is its differential nature. Because of cross

coupling, the two sides of the DNSA give feedback to each other. The 2-parallel NSR-SA version has differential output, but there is no cross-coupling. This means that in presence of noise or BL leakage, both outputs of may be the same. Cancellation of the common mode noise is the big advantage of DNSA.

Data-Valid delay for 16-bit wide SRAM

Cycle Time (ns)	Probability	Saving %
0.88	0.5	17.0
0.89	0.6	16.0
0.9	0.7	15.1
0.92	0.8	13.2
0.95	0.9	10.4
1.06	Worst Case	

Data-Valid delay for 32-bit wide SRAM

Cycle Time (ns)	Probability	Saving %
0.904	0.5	14.7
0.914	0.6	13.8
0.927	0.7	12.5
0.945	0.8	10.8
0.976	0.9	7.9
1.06	Worst Case	

Figure 3.6: Performance benefit of using adaptive delay based sensing scheme in an Asynchronous SRAM. At the 50% point, which is the mean case, performance benefit is 17%. The SRAM size is 512 rows, 16-bit or 32-bit wide, 8 to 1 column mux-ing. Typical corner and worst case across 100K MC simulations, 45nm technology. VDD is 1V.

The numbers in Figure 3.5 are worst case numbers. However, most read operations will be faster, and will generate the Data-Valid signal faster. Figure 3.6 shows the delay of a 512 row, 16-bit and 32-bit wide SRAM along with the probability of that delay. The delay numbers are obtained by utilizing the montecarlo results of the DNSA and NSR-SA. An asynchronous SRAM using a conventional strobed SA would target worst case operation (here we have taken the 4 sigma point), whereas using our DNSA and sensing scheme, we can have a much lower delay, most of the time. The numbers include the delay introduced by the OR and AND gates. The bit-width of the SRAM determines the delay, larger is the bit-width more is the chance of having a

slow bit-cell in the word being read. Thus, by using the DNSA we can achieve up to 17% improvement in the SRAM delay as compared to an SRAM using NSR-SA. In [10], NSR-SA area and conventional strobed SA area was reported $19\mu\text{m}^2$ and $12\mu\text{m}^2$ respectively. Since we use two NSR-SAs in the DNSA, we can estimate our DNSA area to be about 2.5X the NSR-SA. Taking bit-cell area to be $0.25\mu\text{m}^2$ and a 512×256 , 32-bit output SRAM with 8 to 1 column muxing, and 75% array efficiency, on the SRAM level, there is a 2.9% overhead of using NSPS because of the bigger SAs. This is a small area penalty, and is something that's expected because most area in an SRAM is consumed by bit-cells.

3.5. Conclusion

SRAM read delay has traditionally been limited by the worst case bit-cell across the entire array. As technology scaling, local variation increases, and this is making worst case design more and more pessimistic. We propose solutions that help shift away from the worst case paradigm for asynchronous SRAM designs. We show the circuit limitations that prevent conventional sensing schemes from displaying delay that's related to the bit-cell being read. Then, we propose two SA options that help achieve the goal of adaptive delay. We describe a novel differential non-strobed SA (DNSA) that is 8.5% lower in energy per read, and more robust to common mode noise than using a single ended NSR-SA in differential mode. The adaptive sensing scheme using the DNSA achieves delay that is low when the bit-cell is fast and high when the bit-cell is slow. As a result average delay is improved by 17%, as compared to a conventional worst case design SRAM.

Chapter 4: Conclusion and Future Work

Sense amplifiers and sensing schemes play a crucial role in determining power and performance of SRAMs as we have seen in this work. Non-strobed sensing, which is the central theme of this thesis, is a topic that has not been explored much. This chapter summarizes the ideas, implementations and results presented in the thesis, and also sets up a roadmap for future work on non-strobed sensing.

4.1. Summary

In this thesis we explored the advantages of non-strobed sensing for SRAMs. SRAM designers face the challenge of growing local variation as technology scaling continues. Because of variation scaling doesn't provide the automatic performance and power advantages that it traditionally did. Thus, innovative circuit and architectural techniques are required to improve performance and reduce power consumption of SRAMs.

In the first part of the thesis we explored the idea of pipelining an SRAM into two cycles to improve the throughput of instruction caches. SRAM pipelining is tough to implement while using conventional strobed SA as it uses fixed delays to time the SRAM internal operations. Thus, we use non-strobed sensing with a modified timing where we truncate the WL pulse, and trade-off BL droop development time for SA delay. This enables us to split the SRAM read operation into two equal pipeline stages. Our technique helps increase performance by 1.6X as compared to a non-pipelined strobed SRAM, while also decreasing energy per read by 28%. As compared to a 2-stage pipelined strobed SRAM, our technique is 18% faster.

SRAM read delay has traditionally been limited by the worst case bit-cell across the entire array. In the second part of the thesis, we propose solutions that help shift away from the worst case paradigm for asynchronous SRAM designs. We show the circuit limitations that prevent conventional sensing schemes from displaying delay that's related to the bit-cell being read. Then, we propose two SA options that help achieve the goal of adaptive delay. We describe a novel differential non-strobed SA (DNSA) that is 8.5% lower in energy per read, and more robust to common mode noise than using a single ended NSR-SA in differential mode. The adaptive sensing scheme using the DNSA achieves delay that is low when the bit-cell is fast and high when the bit-cell is slow. As a result average delay is improved by 17%, as compared to a conventional worst case design SRAM.

4.2. Contributions

In this thesis we presented new ideas and circuits and demonstrated their benefits for 45nm SRAM. It must be noted that the ideas can be applied to any other memory like DRAM, or emerging memories, and the proposed sense amplifier (DNSA) can be used in interconnects as well. The results are based on simulations in 45nm technology. The following is a list of contributions of this thesis:

- Shows why the long BL droop development phase limits the benefit of pipelining an SRAM
- Shows for the first time how the unique characteristics of a non-strobed sense amplifier can be used to drastically reduce the BL droop development phase at the cost of higher SA delay. This is nothing but a knob that helps balance stages of a pipeline, thereby maximizing the benefit of pipelining an SRAM.

- Demonstrates the implementation of the Non-Strobed Pipelined Sensing (NSPS) scheme. NSPS helps increase performance by 1.6X as compared to a non-pipelined strobed SRAM, while also decreasing energy per read by 28%. As compared to a 2-stage pipelined strobed SRAM, NSPS is 18% faster.
- Shows that conventional (strobed) sensing can only be designed with a worst case timing scheme and thus cannot be used to design an (asynchronous or synchronous) SRAM whose delay depends on the particular bit-cell being read.
- Demonstrates a sensing scheme using a novel differential non-strobed sense amplifier to enable local-variation-adaptive SRAM delay. The technique improves asynchronous SRAM average read delay by up to 17%. The differential non-strobed reduces energy per read operation by 8.5% and is more robust than single ended non-strobed sense amplifiers. The proposed sense amplifier can also be used in synchronous SRAMs.

4.3. Future Work

The work presented in this thesis can be extended the following ways:

- The adaptive sensing scheme for asynchronous SRAMs can be extended to synchronous SRAMs as well. This can be done by using the same circuits as in Figure 2.6. However, instead of waiting for the next cycle, latch the output in one clock period only and give that data to the processor as shown in Figure 4.1. This would mean that most of the times you would latch the correct output, but sometimes the latched value would be incorrect. This would happen when the bit-cell or the SA is slow. To detect such slow read, latch the same output again at the next falling edge, and compare the data with the previously latched data. If both are the same, do nothing because the processor got the correct data. If the two values

differ, then flush the previous cycle, and give the new data to the processor. This would entail a single cycle stall in the processor. There are implementation challenges associated with this scheme, which would be a topic of my research in the future.

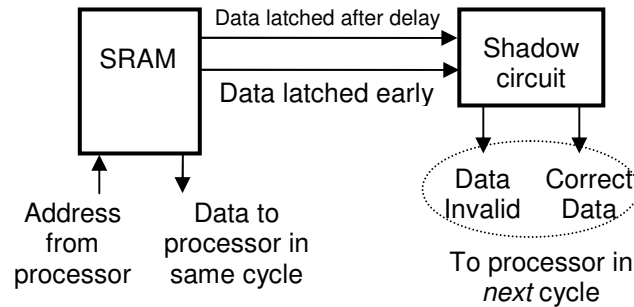


Figure 4.1: A SRAM using an adaptive non-strobed sensing scheme along with a shadow circuit that detects incorrect reads. The data invalid signal and the correct data are made available to the processor available within the one cycle after the read operation.

- Analysis on the best way to implement the capacitors used in the NSR-SA and DNSA. Several options are available, depending on technology like MOS capacitor, metal to metal capacitors, and metal comb capacitors. The capacitor should have lowest parasitics and the highest density. This work would ideally go hand in hand with implementation of a chip demonstrating the ideas of this thesis.

Bibliography

- [1] G. Moore, "The mos transistor as an individual device and in integrated arrays," IRE International Convention Record, vol. 13, pp. 44–52, Mar 1965.
- [2] M. Pelgrom, et al; "Transistor matching in analog CMOS applications, " in IEDM Dig. Tech. Papers, Dec. 1998, pp. 915–918.
- [3] G. Jensen et al., "Monte carlo simulation of semiconductor devices," Computer Physics Communications, pp. 1–61, August 1991.
- [4] Marinissen, E.J.; et al; "Challenges in embedded memory design and test," Design, Automation and Test in Europe, 2005.
- [5] J. Rabaey, et al; "Digital Integrated Circuits: A Design Perspective"
- [6] Yeung, J.; et al; "Robust Sense Amplifier Design under Random Dopant Fluctuations in Nano-Scale CMOS Technologies," SOC Conference, 2006
- [7] A. Agarwal, et al; "Exploring High Bandwidth Pipelined Cache Architecture for Scaled Technology," Design, Automation and Test in Europe, 2003.
- [8] Dhong, S.H.; et al; "A 4.8GHz fully pipelined embedded SRAM in the streaming processor of a CELL processor," ISSCC. 2005
- [9] Lalja, D. J.; "Reducing the branch penalty in pipelined processors," Computer , Jul 1988
- [10] Verma, N.; et al; "A High-Density 45nm SRAM Using Small-Signal Non-Strobed Regenerative Sensing," ISSCC 2008.
- [11] Qazi, M.; et al; "A 512kb 8T SRAM macro operating down to 0.57V with an AC-coupled sense amplifier and embedded data-retention-voltage sensor in 45nm SOI CMOS," ISSCC 2010

- [12] Y. Shakhsheer, **S.Khanna**, K. Craig, S. Arrabi, J. Lach, and B. Calhoun; “A 90nm Data Flow Processor Demonstrating Fine Grained DVS for Energy Efficient Operation from 0.25V to 1.2V,” CICC 2011
- [13] J. Ryan, S.Khanna; et al; “An Analytical Model for Performance Yield of Nanoscale SRAM Accounting for the Sense Amplifier Strobe Signal,” ISQED 2011.

List of Publications

Under Preparation:

- [1] A Fast Low Power Non-Strobed Pipelined Sensing Scheme for Nano-scale SRAMs
(Chapter 2)
- [2] A Fast Low Power Non-Strobed Differential Sense Amplifier and Sensing Scheme for Asynchronous SRAMs (Chapter 3)
- [3] Scheduling and Voltage Selection Tradeoffs in Header-Based Multi Voltage Rail Systems

Published:

- [4] Y. Shakhshier, **S.Khanna**, K. Craig, S. Arrabi, J. Lach, and B. Calhoun; "A 90nm Data Flow Processor Demonstrating Fine Grained DVS for Energy Efficient Operation from 0.25V to 1.2V," CICC 2011 (**Related to Chapter 2**)
- [5] J. Ryan, **S.Khanna**, B. Calhoun; "An Analytical Model for Performance Yield of Nanoscale SRAM Accounting for the Sense Amplifier Strobe Signal," ISLPED 2011 (**Chapter 3**)
- [6] **Khanna, S.**, K. Craig, Y. Shakhshier, S. Arrabi, J. Lach, and B. Calhoun, "Stepped Supply Voltage Switching for Energy Constrained Systems", ISQED, 2011.

- [7] Y. Yao , J. Huang , **S. Khanna** , A. Shelat , B. H. Calhoun , J. Lach , D. Evans, "A Sub-0.5V Lattice-Based Public-Key Encryption Scheme for RFID Platforms in 130nm CMOS," RFIDSec Asia 2011
- [8] Mann, R. W., J. Wang, S. Nalam, **S. Khanna**, G. Bracer, H. Pilo, and B. H. Calhoun, "Impact of circuit assist methods on margin and performance in 6T SRAM", Journal of Solid State Electronics, vol. 54: Elsevier, pp. 1398-1407, 11/2010.
- [9] Calhoun, B. H., **S. Khanna**, Y. Zhang, J. Ryan, and B. Otis, "System Design Principles Combining Sub-threshold Circuits and Architectures with Energy Scavenging Mechanisms", International Symposium on Circuits and Systems (ISCAS), Paris, France, pp. 269-272, 05/2010.
- [10] Calhoun, B. H., S. Arrabi, **S. Khanna**, Y. Shakhsher, K. Craig, J. Ryan, and J. Lach, "REESES: Rapid Efficient Energy Scalable ElectronicS", GOMAC Tech, 03/2010.
- [11] Calhoun, B. H., J. Ryan, **S. Khanna**, M. Putic, and J. Lach, "Flexible Circuits and Architectures for Ultra Low Power", Proceedings of the IEEE, vol. 98, pp. 267-282, 02/2010.
- [12] **Khanna, S.**, and B. H. Calhoun, "Serial Sub-threshold Circuits for Ultra-Low-Power Systems", International Symposium on Low Power Electronics and Design, 8/19/2009.
- [13] Calhoun, B. H., J. Bolus, **S. Khanna**, A. D. Jurik, A. F. Weaver, and T. N. Blalock, "Sub-threshold Operation and Cross-Hierarchy Design for Ultra Low Power Wearable Sensors", International Symposium on Circuits and Systems, 5/24/2009.

- [14] Calhoun, B. H., **S. Khanna**, R. Mann, and J. Wang, "Sub-threshold Circuit Design with Shrinking CMOS Devices", International Symposium on Circuits and Systems, 3/24/2009.